590638

Markov Tracking for Agent Coordination

Richard Washington
Caelum Research
NASA Ames Research Center
MS 269-2
Moffett Field, CA 94035
richw@ptolemy.arc.nasa.gov

Abstract

Partially observable Markov decision processes (POMDPs) are an attractive representation for representing agent behavior, since they capture uncertainty in both the agent's state and its actions. However, finding an optimal policy for POMDPs in general is computationally difficult. In this paper we present *Markov Tracking*, a restricted problem of coordinating actions with an agent or process represented as a POMDP. Because the actions coordinate with the agent rather than influence its behavior, the optimal solution to this problem can be computed locally and quickly. We also demonstrate the use of the technique on *sequential POMDPs*, which can be used to model a behavior that follows a linear, acyclic trajectory through a series of states. By imposing a "windowing" restriction that restricts the number of possible alternatives considered at any moment to a fixed size, a coordinating action can be calculated in constant time, making this amenable to coordination with complex agents.

1 Introduction

Stochastic representations of agents can capture aspects that are otherwise difficult to model, such as errors, alternative outcomes of actions, and uncertainty about the world. These difficulties may arise because of other agents in the world, factors that are simply not well understood, or lack of representation detail.

Markov models [2] are a popular choice for constructing stochastic representations. A Markov decision process (MDP) describes an agent's state as a discrete set of situations. The effects of an agent's actions are represented as probability distributions over the states, reflecting both the range of possible outcomes and their likelihoods. An agent's behavior is a sequence of states and actions. The usual way to plan a behavior in an MDP is to construct a *policy*: a mapping from states to actions, indicating for each state which action the agent will take when it is in that state (note that a policy allows multiple behaviors, since there are multiple possible outcomes of actions). Rewards associated with states and actions specify the local utility of a policy; the total reward of a state for a given policy is simply the sum of the rewards of all the possible behaviors allowed by the policy (weighted by the probabilities). An optimal policy can be computed for an MDP, specifying for each state the optimal action to take to maximize the total utility of the plan.

The classic MDP, however, does not account for uncertainty in the agent's state. Often this state is known only indirectly through observations. If there is inaccuracy or uncertainty in the observation, this indirect information reflects only imprecisely the actual process state. To account for the state uncertainty, MDPs have been extended to partially observable MDPs (POMDPs) [6]. In this model, the underlying agent is an MDP, but the state is only indirectly known; an *observation* is produced on each state transition. The model specifies the probability of seeing an observation in a state (this can be produced in practice by experimental study). Instead of an exact state, the knowledge of the process can be represented as a probability distribution over states, called the *belief state*. A policy in a POMDP is a mapping from belief states to actions. An optimal POMDP policy is thus a mapping from belief states to actions, indicating the optimal action to take in each belief state. This can be useful when the agent has imprecise knowledge of its own state, or an external agent is trying to control (or assist) the agent.

The major drawback of POMDPs is that finding an optimal plan is computationally daunting. The state of the art allows problems of up to about 100 states, and nowhere near real time [5]. Approximation algorithms [15, 8, 13, 14] face a tradeoff that severely compromises solution quality for speed.

In this paper, we look at a restricted problem, *Markov Tracking*, that is concerned with coordinating actions with an agent or process rather than influencing its behavior. It uses the POMDP model to follow the agent's state, and reacts optimally to it. Thus it finds the optimal coordination plan for an external agent. We show that this restriction allows the optimal action to be computed locally, thus allowing the optimal plan to be computed efficiently. Despite the restriction on the model, this approach has a number of interesting possible application areas.

We also discuss the application of Markov Tracking to a subclass of problems called *sequential POMDPs*. Within this class the optimal action is not only computed locally, but can be computed in constant time under certain conditions, allowing true on-line performance with large-scale problems.

To appear in Proceedings of Agents '98: Second International Conference on Autonomous Agents (May 10-13, 1998).
Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

The paper is organized as follows. First we define more precisely the notions of MDPs and POMDPs. Then we describe Markov Tracking and how it applies to POMDPs in general. We then introduce the notion of a sequential POMDP; we show how Markov Tracking with a "windowing" restriction applies to sequential POMDP problems. We present potential applications of the method in the general and restricted cases and some experimental results. Finally we conclude with a discussion of related work, and the promise and limitations of the work.

2 Markov decision processes

In this section we briefly review Markov processes, and in particular POMDPs. We will borrow the notation of [7], adding or changing only as required for the problem at hand; the reader can refer there for a more complete explanation of the framework. The mathematical details, while necessary to prove the optimality of the approach, are not necessary to understand how Markov Tracking works.

We assume that the underlying process, the *core process*, is described by a finite-state, stationary Markov chain. The core process is captured by the following information:

- a finite set $\mathcal{N} \equiv \{1, \ldots, N\}$, representing the possible states of the process
- a variable $X_t \in \mathcal{N}$ representing the state of the core process at time t
- a finite set $\mathcal{A}$ of actions available
- a matrix $P = [p_{ij}], i, j \in \mathcal{N}$ specifying transition probabilities of the core process: $P(a) = [p_{ij}(a)]$ specifies the transition probabilities when action $a \in \mathcal{A}$ is chosen
- a reward matrix $R = [r_{ij}], i, j \in \mathcal{N}$ specifying the immediate rewards of the core process: $R(a) = [r_{ij}(a)]$ specifies the reward received when the action $a \in \mathcal{A}$ is executed, moving the process from state i to state j. We will use the shorthand

$$\varrho_i(a) = \sum_{j \in \mathcal{N}} r_{ij}(a) p_{ij}(a)$$

to denote the reward of taking action a when in state i, and $\varrho(a) = \{\varrho_1(a), \ldots, \varrho_N(a)\}$.

So at time t, the core process is in state $X_t = i$, and if an action $a \in \mathcal{A}$ is taken, the core process transitions to state $X_{t+1} = j$ with probability $p_{ij}(a)$, receiving immediate reward $r_{ij}(a)$.

In MDPs with full observability, actions are chosen by a policy that maps states to actions. The optimal policy is the policy that maximizes the utility of each state. The value of a state under the optimal policy (given full observability) and an infinite planning horizon is defined as:

$$v(i) = \max_{a \in \mathcal{A}} \left\{ \varrho_i(a) + \beta \sum_{k \in \mathcal{N}} v(k) p_{ik}(a) \right\}$$

where $0 \leq \beta < 1$ is a discount factor (this ensures a bounded value function).

For a finite horizon n and full observability, the value function is analogous. We denote the state value under the optimal n-horizon policy as:

$$\begin{aligned} v^0(i) &= r_i(0) \\ v^n(i) &= \max_{a \in \mathcal{A}} \left\{ \varrho_i(a) + \beta \sum_{k \in \mathcal{N}} v^{n-1}(k) p_{ik}(a) \right\}, \text{for } n > 0 \end{aligned}$$

where $r_i(0)$ is the terminal reward received when the process ends up in state i. Note that

$$lim_{n \to \infty} v^n(i) = v(i).$$

However, in a partially observable MDP, the progress of the core process is not known, but can only be inferred through a finite set of observations. The observations are captured with the following information:

- a finite set $\mathcal{M} \equiv \{1, \ldots, M\}$ representing the possible observations
- a variable $Y_t \in \mathcal{M}$ representing the observation at time t
- a matrix $Q = [q_{ij}], i \in \mathcal{N}, j \in \mathcal{M}$ specifying the probability of seeing observations in given states: $Q(a) = [q_{ij}(a)]$, where $q_{ij}(a)$ denotes the probability of observing j from state i when action $a \in \mathcal{A}$ has been taken
- a state distribution variable $\pi(t) = \{\pi_1(t), \ldots, \pi_N(t)\}$, where $\pi_i(t)$ is the probability of $X_t = i$ given the information about actions and observations
- an initial state distribution $\pi(0)$.

At time t, the observation of the core process will be Y_t. If action $a \in \mathcal{A}$ is taken, we can define a function to determine Y_{t+1}. In particular, we define

$$\gamma(j|\pi(t), a) = \sum_{i \in \mathcal{N}} q_{ij}(a) \sum_{k \in \mathcal{N}} p_{ki}(a) \pi_k(t) \qquad (1)$$

as the probability that $Y_{t+1} = j$ given that action $a \in \mathcal{A}$ is taken at time t and the state distribution at that time is $\pi(t)$.

To determine the state distribution variable $\pi(t+1)$, we define the transformation T as follows:

$$\begin{aligned} \pi(t+1) &= T(\pi(t)|j, a) \\ &= \{T_1(\pi(t)|j, a), \ldots, T_N(\pi(t)|j, a)\} \end{aligned}$$

where

$$T_i(\pi(t)|j, a) = \frac{q_{ij}(a) \sum_{k \in \mathcal{N}} p_{ki}(a) \pi_k(t)}{\sum_{l \in \mathcal{N}} q_{lj}(a) \sum_{k \in \mathcal{N}} p_{kl}(a) \pi_k(t)}, \qquad (2)$$

for $i \in \mathcal{N}$, and where $\pi(t)$ is the state distribution at time t, $a \in \mathcal{A}$ is the action taken at that time, resulting in observation $j \in \mathcal{M}$.

Actions are chosen by a decision rule (or plan) that maps state distributions to actions. The utility of a state distribution π under the optimal decision rule can be computed by the POMDP value function:

$$V(\pi) = \max_{a \in \mathcal{A}} \left\{ \pi \cdot \varrho(a) + \beta \sum_{j \in \mathcal{M}} V[T(\pi|j, a)] \gamma(j|\pi, a) \right\}. \qquad (3)$$

For the finite-horizon case, the POMDP value function is again analogous:

$$V^0(\pi) = \pi \cdot r(0) \quad (4)$$

$$V^n(\pi) = \max_{a \in \mathcal{A}} \left\{ \pi \cdot \varrho(a) + \beta \sum_{j \in \mathcal{M}} V^{n-1}[T(\pi|j,a)]\gamma(j|\pi,a) \right\}, \text{for } n > 0 \quad (5)$$

Note that

$$lim_{n \to \infty} V^n(\pi) = V(\pi).$$

3 Markov Tracking

Suppose that instead of wanting to find the optimal plan to control an agent, we wanted to find the optimal plan to coordinate with an agent. In this case we consider actions that do not directly influence the agent's behavior. Instead, what is important is that the correct action is taken with respect to the actual agent state. If the agent is represented by a POMDP, then in fact there is in general uncertainty about the agent's state, so the choice of action must take into account the possibility that the action is in fact not the best for each possible state, but is rather the best for the set of possible states taken together.

In the POMDP formalism, the lack of influence of actions on the underlying process can be stated as the independence of the process and the actions:

$$\forall_{a,a' \in A} \forall_{i,j \in \mathcal{N}} P(i|a,j) = P(i|a',j) \quad (6)$$

$$\forall_{a,a' \in A} \forall_{o \in \mathcal{M}} \forall_{i \in \mathcal{N}} P(o|i,a) = P(o|i,a') \quad (7)$$

Informally, this means that the transition from one state to another is independent of the action, and the observation is also independent of the action. What remains to distinguish one action from another is the reward function.

Note that this is realistic only in the sense that the actions in this framework are in fact external, coordinating actions. One thing that is lost in this representation of the problem is the dependency between the agent's internal actions and its behaviors. When the agent follows a policy internally, this is information that could allow a more accurate (but slower) tracking of its behavior.

Using the independence equations along with the finite-horizon POMDP value equations 4–5, we can see by simple induction over the horizon depth that

$$\forall_{a,a' \in A} \varrho(a,s) \geq \varrho(a',s) \rightarrow V_a(s) \geq V_{a'}(s)$$

that is, that an action with a greater immediate reward will in fact give a higher overall value. This means that the optimal action, which is the action that gives the highest value (see Equation 3), is in fact in this case the action with the greatest immediate reward. In the general POMDP case, this isn't necessarily true because of the difference in transition and observation probabilities.

So what does this model give us? We now have a process (the agent) that transitions probabilistically and provides observations also probabilistically. Using the POMDP transition function (Equation 2), the current belief state $b(s)$ is updated, and the optimal action is the action that maximizes the immediate reward for the belief state. In effect, this relies on the accuracy of the observations and the amount of non-determinism in the underlying process, since no control is exerted to gain more information. But at the same time, maintaining multiple hypotheses about the current state allows the method to remain "on track" even in the presence of noise and uncertainty.

3.1 Applications of Markov Tracking

Despite the apparent simplicity of the approach, there exist a variety of applications for which it appears to be well-suited, including robotics, speech recognition, and music. For example:

- In telerobotics, a (possibly human) controller directs the robot to move around the environment. In return, the sensor information from the robot could be used to warn the operator about potentially hazardous (or beneficial) areas.
- In mobile robotics, while navigating through the environment, a robot may have unused sensors (for example, a camera) that could be directed towards suspected obstacles or other objects of interest.
- In system support for space vehicles, the ground team (or on-board computer) receives information from on-board sensors and must decide what experiments can be run given the current state of the system [16].
- In speech recognition, a computer could be given a text, with the task of coordinating with a spoken version of the text (for example foreign-language subtitling or interactive theater).
- In music, a computer could be given a musical score, with the task of playing the accompaniment, allowing for errors on the part of the performer [3].

The latter three applications involve potentially huge state spaces, and the complexity issues arising from that will be discussed in Section 4.

Although the method is provably optimal for the model we have presented, this does not necessarily translate into high performance. For instance, the loss of information about the agent's individual actions and their results may diminish the effectiveness.

To illustrate the method, we have implemented a telerobotics scenario in a simple mobile-robot simulation environment. A robot moves around in its environment autonomously, and it sends its sensor information to another agent. The agent that receives the sensor information has the task of guessing the robot's position from the sensor information and notifying an operator when the robot reaches a hazardous or beneficial state. This is a simplified instance of the general task of "smart alarms" to ask for human assistance when necessary. The second agent could also be responsible for directing the robot to take certain actions (in lieu of human intervention) or change its operations.

The example world is shown in Figure 1 [5]. The world consists of 23 squares, within each of which the robot may be in one of four orientations (N,S,E,W), leading to a total of 92 possible states of the robot. Observations are defined with respect to the walls around the robot, with a moderate level of noise (e.g., see Figure 2). The observation probabilities are taken from the original definition of the problem used in [5], modified to eliminate a distinguished goal state. Without direct information about the robot actions, the transitions are defined probabilistically, with uncertainty

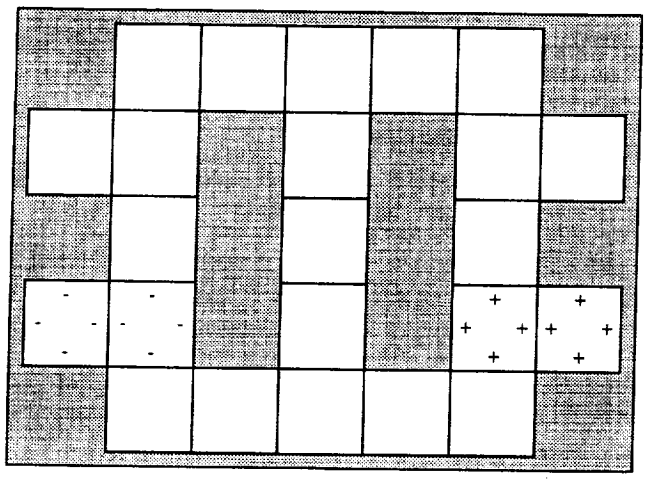

Figure 1: Example world. A state is an orientation (N,S,E,W) within a square. Distinguished states are marked with a '+' (beneficial states) or a '–' (hazardous states).

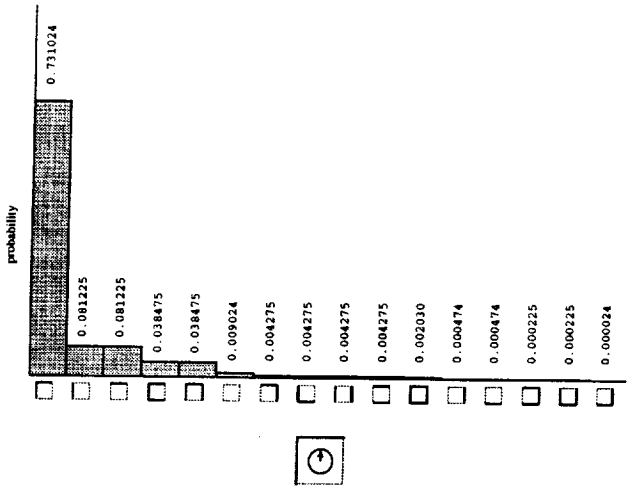

Figure 2: Probability distribution over possible observations for an example state (actual state shown below the distribution).

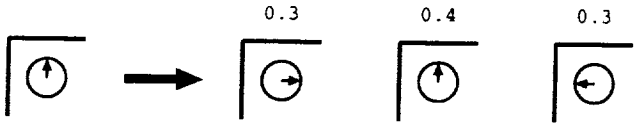

Figure 3: Transition probabilities for an example state.

about the next state based on the lack of knowledge about the robot's movement (e.g., see Figure 3).

Within the world, we defined a set of 8 states (2 squares) marked with a '–' that represent hazardous states, and 8 states (2 squares) marked with a '+' that represent beneficial states. Together these 16 states will be called *distinguished states*. The correct action in a hazardous state is to turn on a red light, and the correct action in a beneficial state is to turn on a green light. Note that because of the symmetry within the world (the differences in rewards for the distinguished states do not show up in the observations), pairs of states (reflections about the center of the world) are completely indistinguishable.

The rewards were defined as follows:

true positive : turning on the correct light in a distinguished state. +2

false positive : turning on a light in a non-distinguished state. -1

false negative : doing nothing in a distinguished state. -2

true negative : doing nothing in a non-distinguished state. +1

Turning on the incorrect light in a distinguished state is in fact a false negative for the light that should have been turned on and a false positive for the light that was actually turned on; it is given a penalty of -2, the maximum penalty of the two. In fact, the only important feature of the penalties is that the reward for a true positive be greater than the penalty for a false positive. The reason is that in this world, the symmetry implies that there will always be at most 50% certainty of being in a distinguished state, and less because of noise. With equal weights for true positives and false positives, the tracking system would never choose to turn on a light.

An experimental trial consisted of the robot taking a random walk around the environment, respecting the underlying probabilistic model in its actions and observations (to avoid states or observations that fall outside the model), but hidden from the Markov Tracking system except for the observations provided. The starting probability distribution was uniform over all states, so that the tracking system started with no information about the robot position. Each trial was run for 1000 iterations (i.e., a random walk of 1000 actions).

The complete experiment consisted of 1000 trials. The results can be seen in Figure 4. Each set of 10 iterations was grouped together to show the behavior over time while reducing the experimental noise. We can see that the rate of identification becomes high after a small number of iterations and remains there. In general, the data show that the system starts with no knowledge and thus generally chooses to do nothing, but that after seeing a series of observations, the certainty grows to where in the great majority of cases, the correct action is taken (the remainder of the errors being largely due to the inherent noise in the model).

The symmetry of the problem precludes a definitive identification of the robot position, even given perfect information. To determine the effect of symmetry on the results, we constructed a version of the problem with one square removed, thus making it asymmetric (see Figure 7). With perfect information the robot would be able to (eventually) distinguish among the states. We would expect this to help reduce the problem of false positives. In fact this is the case, as can be seen from Figure 5. This shows that a small

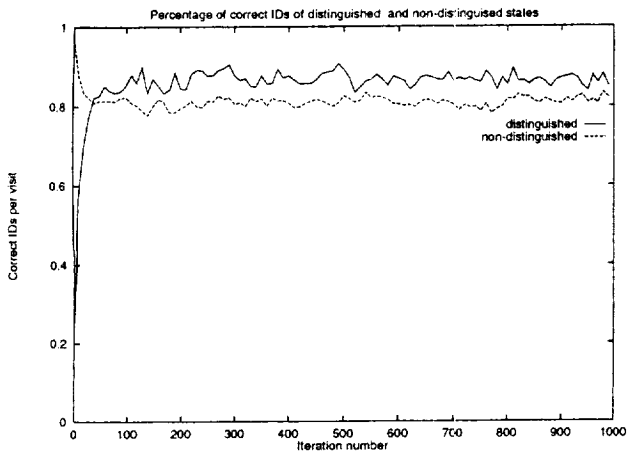

Figure 4: Proportion of true positives per distinguished state and true negatives per non-distinguished state.

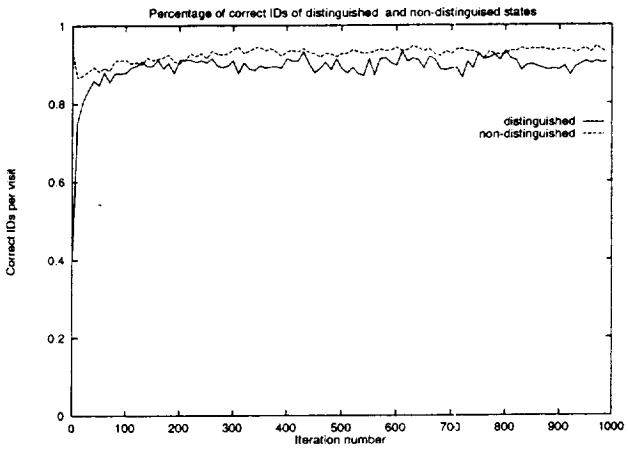

Figure 5: Proportion of true positives per distinguished state and true negatives per non-distinguished state. Asymmetric world.

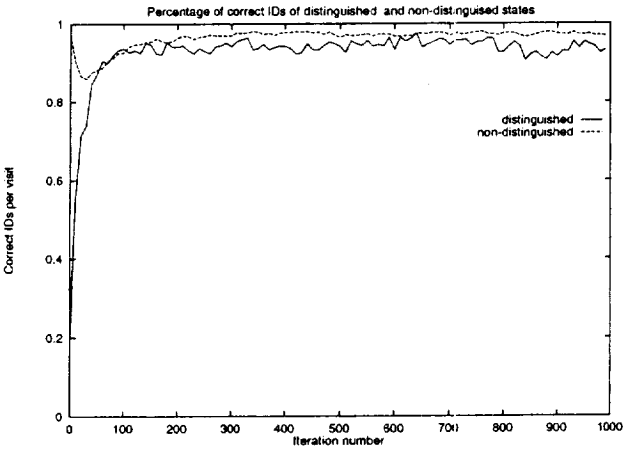

Figure 6: Proportion of true positives per distinguished state and true negatives per non-distinguished state. World with marked state.

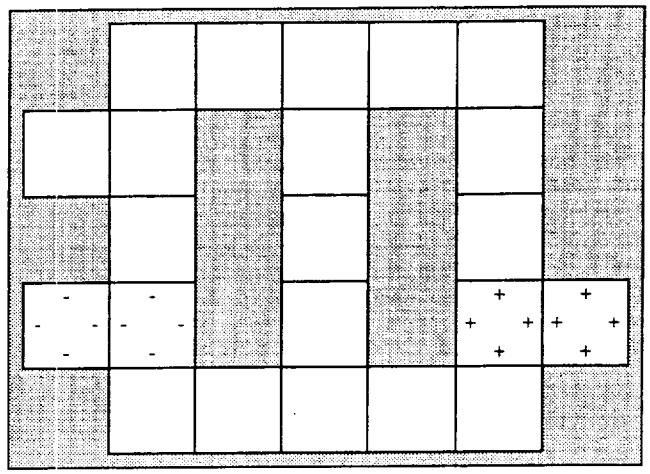

Figure 7: Example world, with one square removed to make it asymmetric.

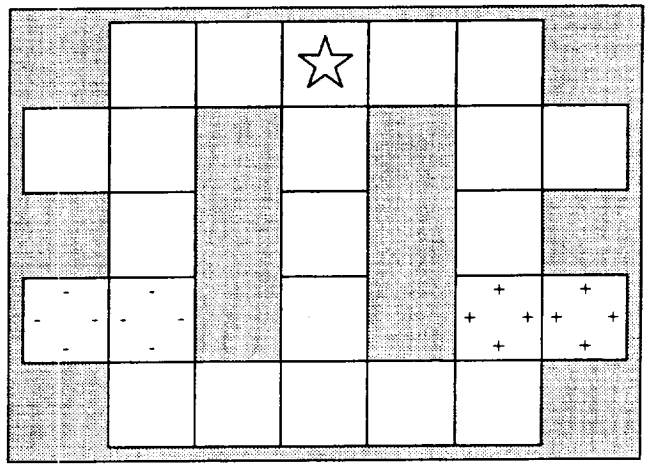

Figure 8: Example world, with a marker added (the square with the star).

change in the world can make a significant difference in the accuracy of identification.

The rate of recognition grows even higher when a marker is placed in one square of the world that tells the robot that it is in that square when it happens to pass through (see Figure 8). The rotational uncertainty remains, since the orientation is unspecified. The presence of the marker allows the robot to localize itself from time to time and thus nearly eliminate errors (see Figure 6). It is interesting that an occasional beacon in a noisy world allows this level of improvement.

In all of these cases, the true transition model of the underlying agent (the transition probabilities of each possible action in each state) is replaced by an action-independent transition model, thus losing information. So the fact that the agent state can be effectively tracked even with this loss of information is an encouraging result.

4 Markov Tracking for Sequential POMDPs

The computational complexity of choosing the optimal action in Markov Tracking is $\mathcal{O}(N)$, since it involves a simple

vector dot product of the belief state with a reward vector. For most applications, even with relatively large state spaces, this poses little problem for real-time performance. However, hidden behind this is the belief state update, which involves $\mathcal{O}(N^2)$ calculations (see Equation 2). In the example problem shown in this paper, a trial of 1000 iterations takes less than a half minute on a Sun SparcStation 20. But for very large state spaces, the complexity could start to adversely affect on-line performance. In this section we show how a restricted class of POMDPs can be tracked in constant time. This allows complex agents to be tracked.

4.1 Sequential POMDPs

A *sequential POMDP* is a restricted POMDP in which the state transitions in the underlying MDP are constrained to the same state or the "following" state:

$$\forall_{i,j \in \mathcal{N}} \forall_{a \in \mathcal{A}} p_{ij}(a) > 0 \rightarrow i \leq j \leq i+1$$

Graphically such a model looks like Figure 9. In fact, this can be generalized a bit: if the number of nonzero transitions from any state is less than a constant bound, the results in this section will hold. However, in the remainder of the section, we will hold to the more restrictive definition for ease of explanation and understanding.

Given the restriction on the model, the state distribution update given in Equation 2 can be computed in $\mathcal{O}(N)$, since the denominator can be computed once for all T_i at a cost of $\mathcal{O}(N)$ and the numerator is computed separately for each T_i at constant cost for each (thus $\mathcal{O}(N)$ total). In addition, the memory required for the transition and reward matrices is $\mathcal{O}(N \cdot |\mathcal{A}|)$ and the memory for the observation matrix remains $\mathcal{O}(N \cdot |\mathcal{M}| \cdot |\mathcal{A}|)$. For Markov Tracking, the $|\mathcal{A}|$ terms disappear, since the transitions are independent of the actions.

4.2 Windowing

In POMDPs, the state of knowledge of the current state of the process is represented by the belief state, which is a probability distribution over the set of states. Over time, this distribution may have many non-zero but vanishingly small elements, each of which must be taken into account when updating the belief state. However, if we limit the distribution to the k most probable states, for some constant k, by zeroing the rest and renormalizing the distribution, the computational complexity of tracking in the general POMDP case reduces from $\mathcal{O}(N^2)$ for N states to $\mathcal{O}(N)$, based on the cost of the belief state update in Equation 2.

4.3 On-Line Markov Tracking

For sequential POMDPs, given a distribution of k possible states, there are at most $2k$ states that could have a nonzero probability in the following time step (of which k will be retained). This means that the belief state update can be limited to those states, and in fact that makes the belief state update $\mathcal{O}(1)$ (constant).

Note however that there is some risk in limiting the belief state distribution to a fixed-size window. First, the coordination action chosen will be the optimal only with respect to the truncated predictions. In particular, in the worst case, the truncated belief state could correspond only remotely to the "real" belief state.

Perhaps a more troubling problem, for both theory and practice, is that it is possible for an observation to be seen that is in fact impossible in any of the $2k$ successor states (because it comes from a state that was very improbable but in fact correct). This results in a null state distribution when the formulas are used.

We propose the following method to handle this (fortunately rare) occurrence. The transition probabilities indicate the evolution of the belief state given no observations, so the tracking process falls back on these in the presence of "impossible" observations. This allows the model of the process to evolve over time even in this case. It is a bit like "flying blind" when all the instruments go dead—the best strategy is to continue with the best existing model. In our experience this allows the process to re-orient itself quickly. See Section 6 for other ideas under development.

4.4 Applications of On-Line Markov Tracking

The On-Line Markov Tracking approach can be applied to a number of problem domains. Sequential POMDPs are appropriate for domains that follow a trajectory over time. For example, the computer could have the job of following a spoken text and producing a subtitled text to accompany it. Or the computer could have the job of following a musical score and playing an accompaniment. The generalization of sequential POMDPs to POMDPs with a constant number of transitions per state leads to application domains that include telerobotics and space vehicle status tracking.

We are investigating these and other "real" applications, but for initial results, we constructed a number of randomly-generated scenarios to illustrate our ideas.

Each scenario consisted of a sequence of states of length 250 (plus a distinguished Start and End state). Each state was chosen from a set of 10 possible states, S0–S9. The observations, O0–O9, correspond to the states, but with some noise: the "correct" observation is given with a probability depending on the scenario (described more below), with the rest of the probability distributed on surrounding observations (where surrounding was computed syntactically—the closest observations to O4 are O3 and O5, for example).

A process started in the Start state and transitioned probabilistically from state to state, producing an observation computer randomly according to the noise probabilities. When the process reached the End state, it remained there (i.e., the only transition was to the same state). The Start and End states produced distinguished observations.

Two variants of this scenario were explored: relatively high reliability of transitions and observations, and relatively low reliability. The high reliability scenario used a transition probability of 0.9 of moving to the next state, and a probability of a correct observation in the range 0.825–0.91. The low reliability scenario used a transition probability of 0.7 of moving to the next state, and a probability of 0.635–0.815 of a correct observation. So in the low reliability scenario, the information available is less helpful for determining the precise state. In all cases the start and end states are identified with complete certainty—the interesting question is how well it follows the intermediate states.

With each scenario, Markov Tracking was run with a window of 10 states, which is an admittedly arbitrary choice, but chosen to be both not trivially small (for accuracy) and not too large (for computation). Each experimental trial consisted of running the tracking method for 400 iterations on this process (by which time in almost all cases the process was in the End state). For each of a set of 10 randomly-generated cases within each scenario, we ran 100 trials, so 1000 trials in all for each scenario.

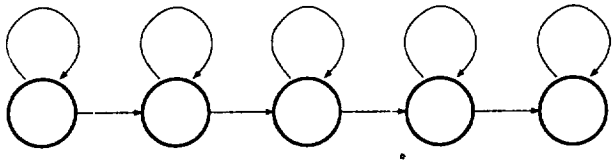

Figure 9: Sequential POMDP. All transitions are to the same or following state.

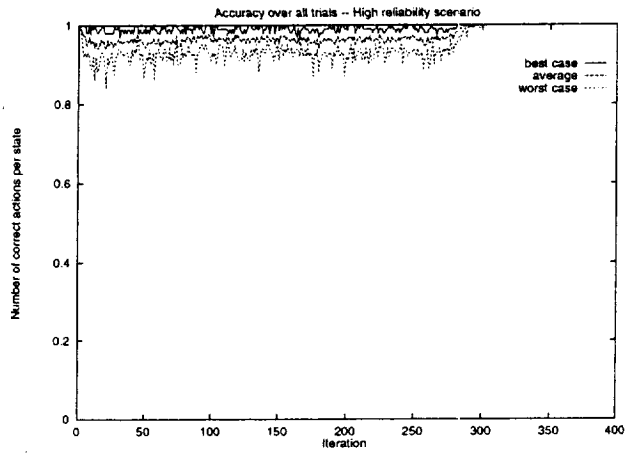

Figure 10: Number of correct identifications per state, high-reliability scenario.

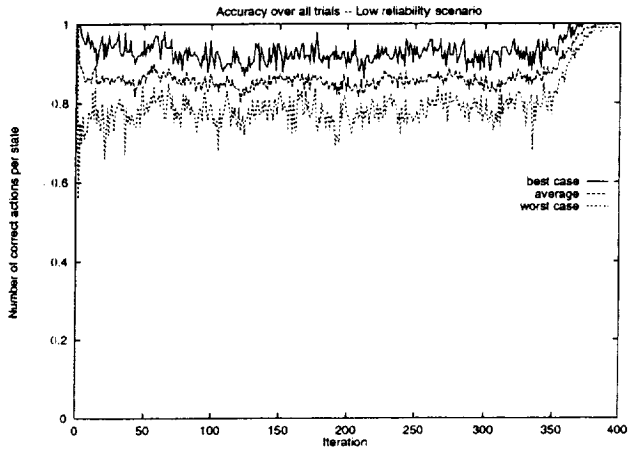

Figure 11: Number of correct identifications per state, low-reliability scenario.

The results can be seen in Figures 10–11. The 100 trials from each case were averaged, and then these 10 averages (one per case) were again averaged to produce the line labeled "average" in the graphs. This is the average over all trials within a scenario. The "worst case" line is the minimum value of the 10 cases, the "best case" line shows the maximum value of the 10 cases. Obviously for any single trial the accuracy for a single iteration is simply 0 or 1, but the results give an idea of the expected accuracy within and across scenarios. The accuracy rises to perfect once all behaviors have been terminated (depending on the actual trial, the length of the behavior could be different).

In the high-reliability scenario, the average accuracy is at least 94%, with a median of 96.7%. In the low-reliability scenario, the average accuracy is at least 80%, with a median of 85.9%. This shows that, as expected, accuracy declines with an increase in noise and unpredictability, but that in general the tracking method is able to accurately follow the process. In fact, the errors seem to be mostly local errors caused by incorrect observations. There were no cases of null distributions caused by the truncated probability distributions in the cases examined; in fact, the windowing lost at most 0.0001 of the distribution even in the low-reliability scenario.

5 Related Work

Markov Tracking can be seen as a variant of plan recognition [9, 1], where the goal is restricted to finding the current state of the agent rather the entire plan structure. As a state recognizer, it is also related to mode identification for fault recovery [16].

Markov Tracking is a special case of the general POMDP problem, which is computationally intractable in its full form [5]. Approximation algorithms for the full POMDP problem [15, 8, 10, 13, 14] remain far from real time.

The RESC approach [12, 11] is designed for real-time agent tracking. That approach does not reason explicitly about the probability of multiple possible agent states, but rather commits to one possible state and relies on fast backtracking to switch among possibilities. This backtracking, although restricted, lacks the complexity guarantees that Markov Tracking offers. In addition, the commitment to a single agent state at a time precludes choosing a collaborative action that is the best given the lack of information about the precise state.

The Markov Tracking approach is in some sense complementary to the Kalman filtering approach [4], used widely in robotic applications. The Kalman filter maintains a best estimate of the position, along with an error estimate. The Kalman filter assumes Gaussian noise and error, which is not always the case in the domains discussed here. In robotic navigation, if the robot approaches a branch, the Markov Tracking approach can represent the two discrete possibilities explicitly. The Kalman filter, on the other hand, is a better choice for continuous spaces where its assumptions hold.

6 Discussion

We have introduced the method of Markov Tracking, which chooses the optimal coordination action with respect to an agent modeled as a POMDP. Furthermore, we have shown that the choice of optimal action can be calculated locally, avoiding the lengthy (and for practical purposes, intractable) computation of the general POMDP case. In our initial results, the method performs well in the presence of noise and uncertainty.

We have also introduced On-Line Markov Tracking, a specialization of Markov Tracking that chooses coordination actions for the restricted case of sequential POMDPs with a fixed-size window on the belief state. Under these restrictions, the choice of the coordination action can be calculated locally in constant time per step. This allows the approach to scale up to coordination problems for agent models with huge state spaces.

The power of Markov Tracking rests on its ability to take the best action with respect to the uncertainty of the precise state. So if the sensors give highly reliable information, the best action for the actual state will be taken, but if the state is known only imprecisely, the action that has the best utility over all the possible states is taken. Given the fact that the precision of sensors and tests is increasing over time, it is reasonable to use the estimates given by the sensor and test data. However, it would be a grave error to discount completely the inherent error, and the approach accounts for that as well.

The tracking method remains "on course" even in the presence of noise and a restricted belief state. The size of the window can be altered to handle increasing levels of noise, with of course an accompanying increase in the constant factor of computation.

The problem of 0-probability observations can be vexing. Although in our experiments we didn't see any such cases, in operational systems such a fault couldn't be ignored: flying blind for more than a small number of steps could have disastrous consequences. The issue comes down to whether to believe your observations or your predictions. This could depend on sensor reliability and how much of the prediction distribution was lost by windowing. We are currently investigating ways of "mixing" sensor-based and prediction-based belief states to provide a robust method for recovering from 0-probability observations. Initial results indicate that in practice this will reduce the accumulated error in the truncated belief state.

References

[1] J. F. Allen, H. A. Kautz, R. N. Pelavin, and J. D. Tenenberg. *Reasoning about Plans*. Morgan Kaufmann Publishers, Inc., 1991.

[2] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.

[3] R. Dannenberg. An on-line algorithm for real-time accompaniment. In *Proceedings of the 1984 International Computer Music Conference*, pages 193–198. Computer Music Association, 1984.

[4] A. Gelb. *Applied Optimal Estimation*. M.I.T. Press, 1980.

[5] M. L. Littman, A. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In A. Prieditis and S. Russell, editors, *Proceedings of the Twelfth International Conference on Machine Learning*, pages 362–370, San Francisco, CA, 1995. Morgan Kaufmann.

[6] W. S. Lovejoy. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research*, 28:47–65, 1991.

[7] G. E. Monahan. A survey of partially observable Markov decision processes: Theory, models, and algorithms. *Management Science*, 28(1):1–16, 1982.

[8] R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of IJCAI-95*, 1995.

[9] A. S. Rao. Means-end plan recognition: Towards a theory of reactive recognition. In *Proceedings of KR-94, the Fourth International Conference on Principles of Knowledge Representation and Reasoning*, 1994.

[10] R. Simmons and S. Koenig. Probabilistic robot navigation in partially observable environments. In *Proceedings of IJCAI-95*, 1995.

[11] M. Tambe. Tracking dynamic team activity. In *Proceedings of AAAI-96*, 1996.

[12] M. Tambe and P. S. Rosenbloom. RESC: An approach for real-time, dynamic agent tracking. In *Proceedings of IJCAI-95*, 1995.

[13] R. Washington. Incremental Markov-model planning. In *Proceedings of TAI-96, Eighth IEEE International Conference on Tools With Artificial Intelligence*, 1996.

[14] R. Washington. BI-POMDP: Bounded, incremental partially-observable markov-model planning. In *Proceedings of ECP'97, the Fourth European Conference on Planning*, 1997.

[15] C. C. White, III. A survey of solution techinques for the partially observed Markov decision process. *Annals of Operations Research*, 32:215–230, 1991.

[16] B. C. Williams and P. P. Nayak. A model-based approach to reactive self-configuring systems. In *Proceedings of AAAI-96*, pages 971–978, 1996.